Design Pattern Study Notes

@author Count Monte Cristo

@create D20151223

@version D20151223

@description 记录设计模式（**设计标准，最佳实践**）重要的知识

@motto VICTORY LOVES PREPARATION

@criteria 凯旋基诺

1. @reference <http://www.runoob.com/design-pattern/design-pattern-tutorial.html>
2. **设计模式原则**
   1. **图示**



1. **设计模式分类**
   1. **共有23中模式**
   2. **具体图示**
   3. **分类**
      1. **创建型模式（Creational Patterns）**
         1. **工厂模式（Factory Pattern）**
            1. 意图

（通过Factory）统一创建对象

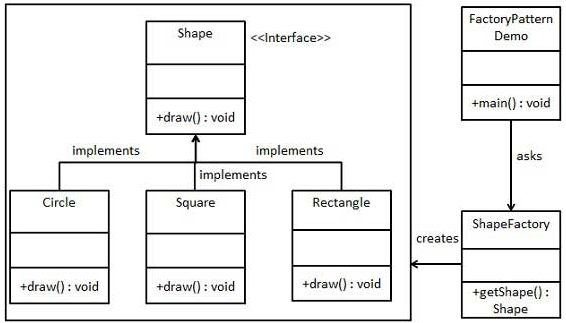
* + - * 1. 优点

**屏蔽产品的具体实现，调用者只关心产品的接口**

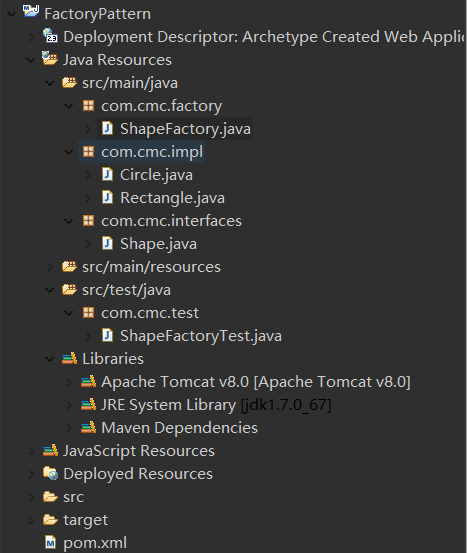
扩展性高，如果想增加一个产品，只要扩展一个工厂类就可以

* + - * 1. 缺点

**每次增加一个产品时，都需要增加一个具体类和对象实现工厂**，使得系统中类的个数成倍增加，在一定程度上增加了系统的复杂度，同时也增加了系统具体类的依赖

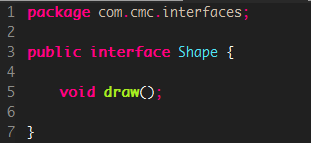
* + - * 1. UML
        2. 实例

程序结构

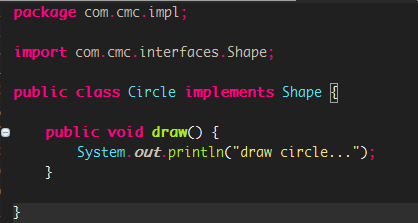


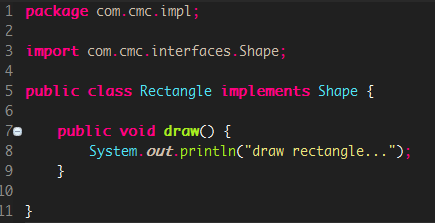
具体内容

Shape接口

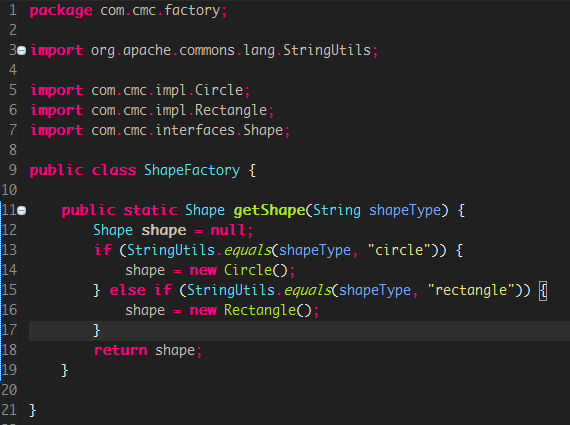


Shape接口具体实现

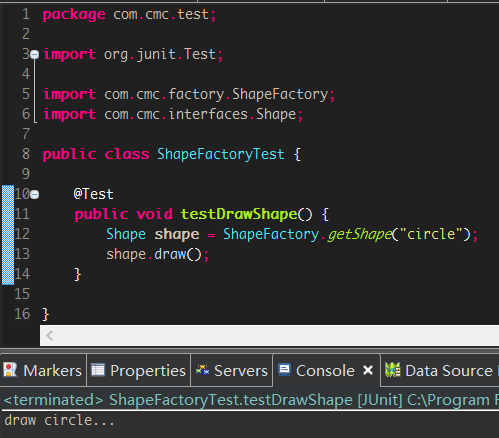




工厂



测试方法以及结果



* + - * 1. 注意

一般来说工厂方法为类所拥有而不是对象，即方法一般是静态的

* + - 1. **抽象工厂模式（Abstract Factory Pattern）**
         1. 意图

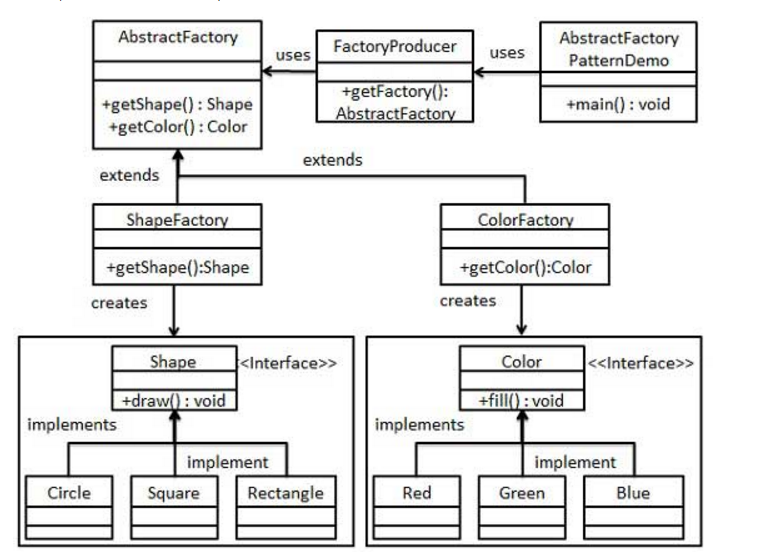
创建工厂

* + - * 1. 优点

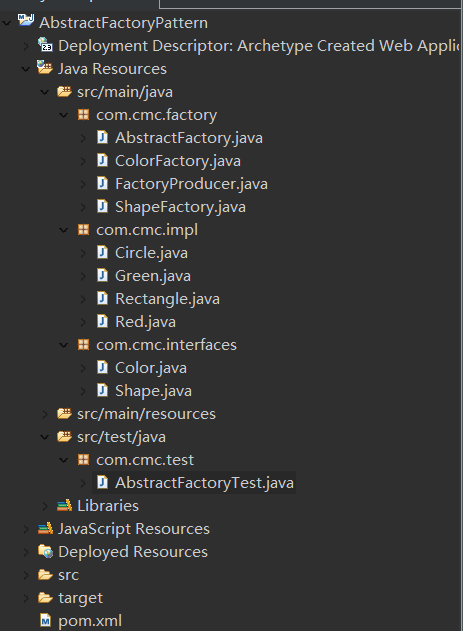
当一个产品族中的多个对象被设计成一起工作时，它能保证客户端始终只使用同一个产品族中的对象

* + - * 1. 缺点

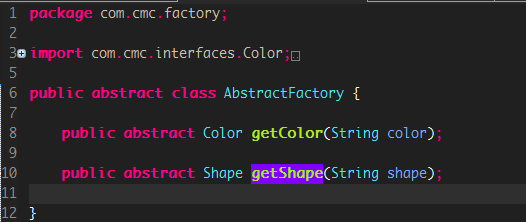
产品族扩展非常困难，要增加一个系列的某一产品，既要在抽象的Producer里加代码，又要在具体的里面加代码

* + - * 1. UML
        2. 实例

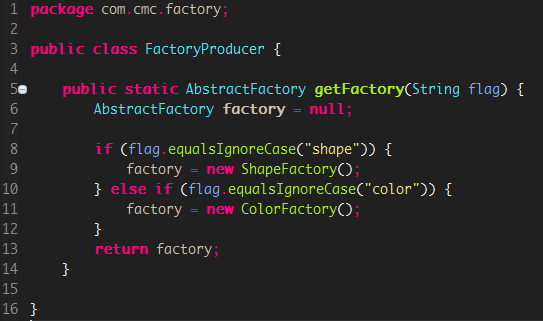
程序结构

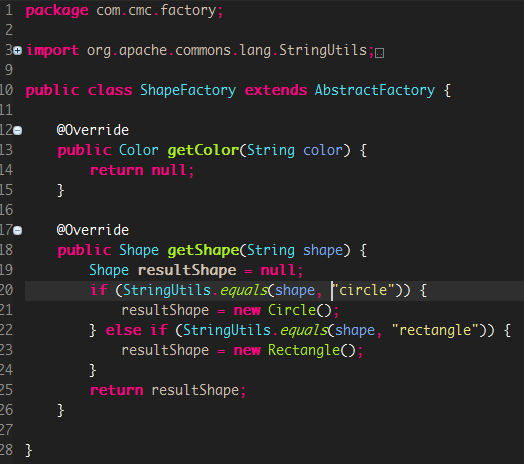


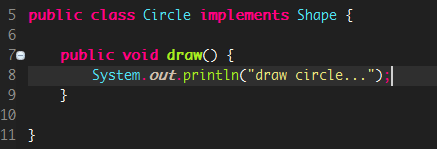
具体内容

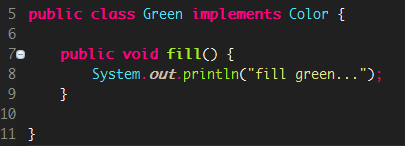


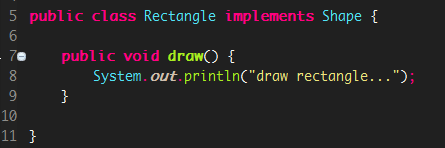


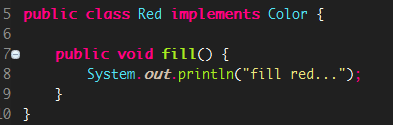


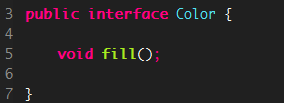


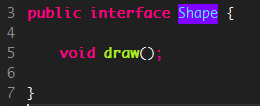


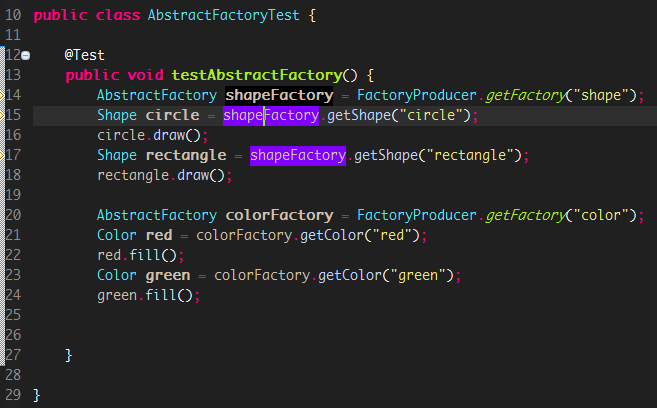


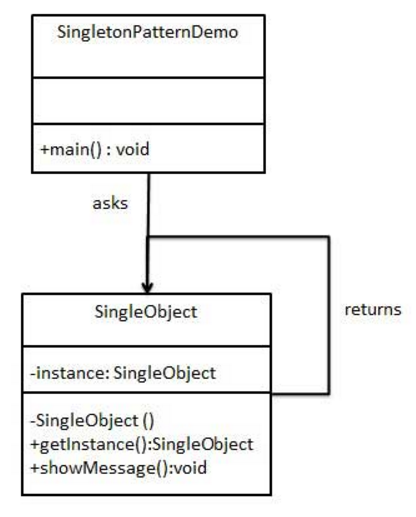




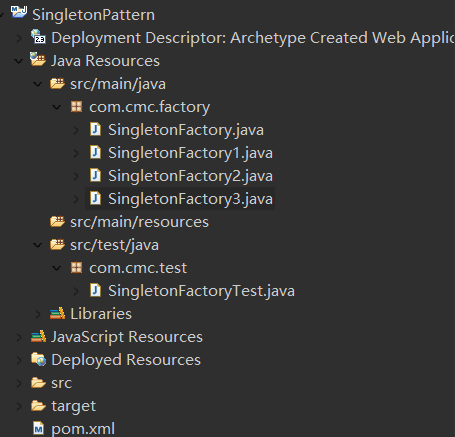




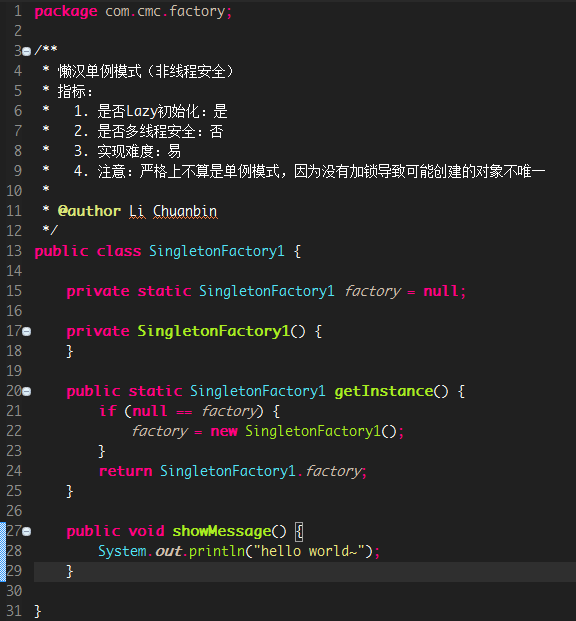


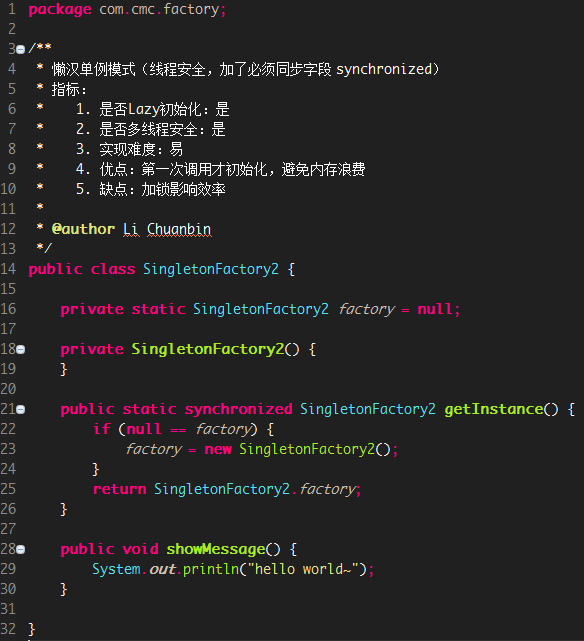
* + - 1. **单例模式（Singleton Pattern）**
         1. 意图
         2. 优点
         3. 缺点
         4. UML
         5. 实例

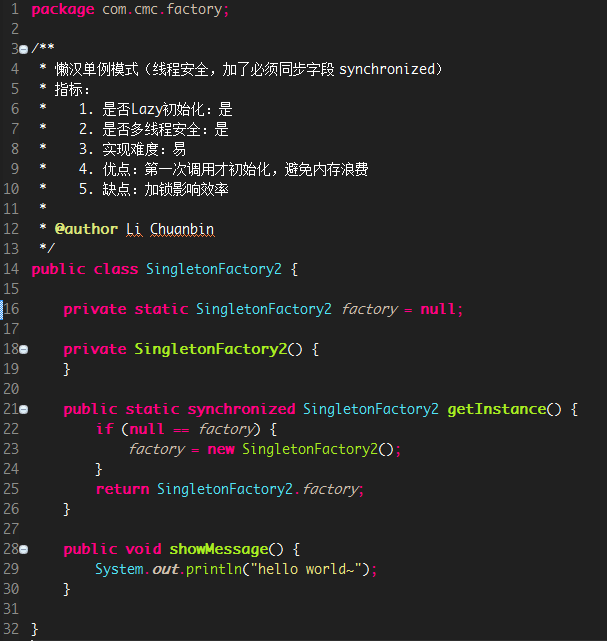
程序结构

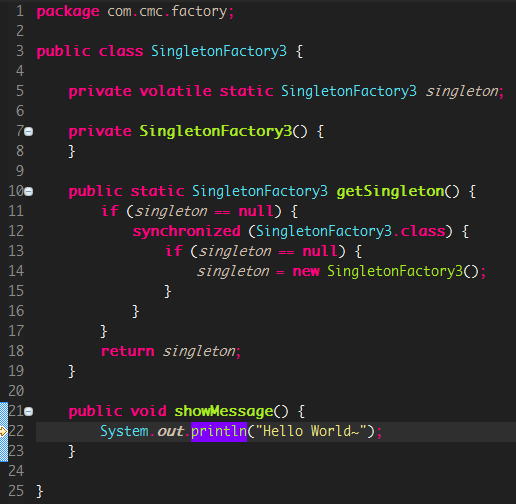


具体内容









* + - * 1. 注意

具体需要注意的见代码注释

* + - 1. **建造者模式（Builder Pattern）**
         1. 意图
         2. 优点
         3. 缺点
         4. UML
         5. 实例
         6. 注意
      2. **原型模式（Prototype Pattern）**
         1. 意图
         2. 优点
         3. 缺点
         4. UML
         5. 实例
         6. 注意
    1. **结构型模式（Structural Patterns）**
       1. **适配器模式（Adapter Pattern）**
          1. 意图
          2. 优点
          3. 缺点
          4. UML
          5. 实例
          6. 注意
       2. **桥接模式（Bridge Pattern）**
          1. 意图
          2. 优点
          3. 缺点
          4. UML
          5. 实例
          6. 注意
       3. **过滤器模式（Filter Pattern）**
          1. 意图
          2. 优点
          3. 缺点
          4. UML
          5. 实例
          6. 注意
       4. **组合模式（Composite Pattern）**
          1. 意图
          2. 优点
          3. 缺点
          4. UML
          5. 实例
          6. 注意
       5. **装饰器模式（Decorator Pattern）**
          1. 意图
          2. 优点
          3. 缺点
          4. UML
          5. 实例
          6. 注意
       6. **外观模式（Facade Pattern）**
          1. 意图
          2. 优点
          3. 缺点
          4. UML
          5. 实例
          6. 注意
       7. **享元模式（Flyweight Pattern）**
          1. 意图
          2. 优点
          3. 缺点
          4. UML
          5. 实例
          6. 注意
       8. **代理模式（Proxy Pattern）**
          1. 意图
          2. 优点
          3. 缺点
          4. UML
          5. 实例
          6. 注意
    2. **行为型模式（Behavioral Patterns）**
       1. **责任链模式（Chain of Responsibility Pattern）**
          1. 意图
          2. 优点
          3. 缺点
          4. UML
          5. 实例
          6. 注意
       2. **命令模式（Command Pattern）**
          1. 意图
          2. 优点
          3. 缺点
          4. UML
          5. 实例
          6. 注意
       3. **解释器模式（Interpreter Pattern）**
          1. 意图
          2. 优点
          3. 缺点
          4. UML
          5. 实例
          6. 注意
       4. **迭代器模式（Iterator Pattern）**
          1. 意图
          2. 优点
          3. 缺点
          4. UML
          5. 实例
          6. 注意
       5. **中介者模式（Mediator Pattern）**
          1. 意图
          2. 优点
          3. 缺点
          4. UML
          5. 实例
          6. 注意
       6. **备忘录模式（Memento Pattern）**
          1. 意图
          2. 优点
          3. 缺点
          4. UML
          5. 实例
          6. 注意
       7. **观察者模式（Observer Pattern）**
          1. 意图
          2. 优点
          3. 缺点
          4. UML
          5. 实例
          6. 注意
       8. **状态模式（State Pattern）**
          1. 意图
          2. 优点
          3. 缺点
          4. UML
          5. 实例
          6. 注意
       9. **空对象模式（Null Object Pattern）**
          1. 意图
          2. 优点
          3. 缺点
          4. UML
          5. 实例
          6. 注意
       10. **策略模式（Strategy Pattern）**
           1. 意图
           2. 优点
           3. 缺点
           4. UML
           5. 实例
           6. 注意
       11. **模板模式（Template Pattern）**
           1. 意图
           2. 优点
           3. 缺点
           4. UML
           5. 实例
           6. 注意
       12. **访问者模式 （Visitor Pattern）**
           1. 意图
           2. 优点
           3. 缺点
           4. UML
           5. 实例
           6. 注意
    3. **J2EE模式**
       1. **MVC模式（MVC Pattern）**
          1. 意图
          2. 优点
          3. 缺点
          4. UML
          5. 实例
          6. 注意
       2. **业务代表模式（Business Delegate Pattern）**
          1. 意图
          2. 优点
          3. 缺点
          4. UML
          5. 实例
          6. 注意
       3. **组合实体模式（Composite Entity Pattern）**
          1. 意图
          2. 优点
          3. 缺点
          4. UML
          5. 实例
          6. 注意
       4. **数据访问对象模式（Data Access Object Pattern）**
          1. 意图
          2. 优点
          3. 缺点
          4. UML
          5. 实例
          6. 注意
       5. **前端控制器模式（Front Controller Pattern）**
          1. 意图
          2. 优点
          3. 缺点
          4. UML
          5. 实例
          6. 注意
       6. **拦截过滤器模式（Intercepting Filter Pattern）**
          1. 意图
          2. 优点
          3. 缺点
          4. UML
          5. 实例
          6. 注意
       7. **服务定位器模式（Service Locator Pattern）**
          1. 意图
          2. 优点
          3. 缺点
          4. UML
          5. 实例
          6. 注意
       8. **传输对象模式（Transfer Object Pattern）**
          1. 意图
          2. 优点
          3. 缺点
          4. UML
          5. 实例
          6. 注意